Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem. It is particularly useful when a problem can be broken down into similar sub-problems, each of which can be solved independently using the same approach.

For example, in the context of financial forecasting, calculating compound growth for a number of years is naturally recursive. Each year's future value depends on the value from the previous year, making it suitable for a recursive solution.

Time Complexity

The recursive function makes one call for each year. Therefore, the time complexity is O(n), where n is the number of years. This is considered linear time complexity, as the number of recursive calls grows linearly with the input size (in this case, the number of years).

Optimization Discussion

Memoization is a common optimization strategy in recursive algorithms where intermediate results are stored to avoid redundant computations. However, in this financial forecasting scenario, each recursive call is based on a different value of the amount, and there is no repetition of the same subproblems. Hence, memoization offers no benefit here.

Instead, if performance is a concern, especially for a very large number of years, an **iterative approach** is generally preferred. It uses a simple loop to accumulate the result without the overhead of function calls, making it more efficient in terms of both speed and memory usage.